

打开git bash



在git里面输入命令行

1,安装fekit

(作用：fekit用来模拟服务器（和apache功能一样）)

cnpm install fekit -g 或npm install fekit -g

2，开启fekit服务

fekit server

这里开启的服务的端口号默认是80的。

3，如果想改变端口号

fekit server -p 8090

服务器的端口就改为8090了。

localhost:8090

这里的fekit这个服务的开发是由nodejs开发的。还有

gulp，gulp-server插件等等也是由nodejs开发的。

4，nodejs可以实现服务器的功能，服务器端最基本的功能

是读和写（I/O）;

nodejs全面支持ECMA6.

<https://nodejs.org>是nodejs的官网，里面有nodejs的开发文档

5.，gulp的安装。

首先：gulp的初始化。gulp init

生成一个package.json文件

{

"name": "course-node", //这是项目名

"version": "1.0.0", //项目版本

"description": "", //项目描述

"main": "index.js",

"scripts": { //简化运行时候的指令。

"test": "echo \"Error: no test specified\" && exit 1"，

“main”:"supervisor server.js"

},

"author": "",

"license": "ISC",

"devDependencies": {

"gulp": "^3.9.1"

}

}

6，gulp是一个前端工程化的工具。gulp可以实现读写，编译sass，压缩图片等功能。就是实现方便对文件的归纳管理。

7，模块化开发的标准commonJS,

定义模块：如：name.js

var aaa = {

name : "tomcat",

talk : function(){

return this.name;

}

}

暴露模块（也就是模块的接口）;

module.exports = aaa;

引入模块:request.js

const aaa = require('模块的路径');

如果模块的路径只写模块名，那么程序就会先去node\_modules文件夹里面去找模块，找不到在去找其他文件夹

中的模块。所以要把模块的路径写完整，同级目录写./。上级写../。

应用模块的方法：aaa.talk();

如果在网页里面把引入模块的js文件引入html网页里。

<script src="request.js"></script>

网页就会报错，require is not define。

因为commonjs是基于工程化的工具。require脱离了可以解析的环境，所以报错；require基于gulp是可以的，node环境也可以。想看运行的结果，可以在node环境下测试，

clipboard.png

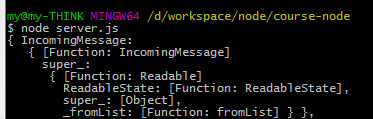
5，用nodejs实现一个server

首先引入http模块，因为http模块在安装node的时候已经安装好了，直接把模块引入过来就可以了。

const http = require("http");

console.log(http);

我们在node环境运行一下，就可以看到http的功能，



在这些功能里面有一个createServer()方法，可以开启一个模拟服务器的功能。

http.createServer(function(request,response){

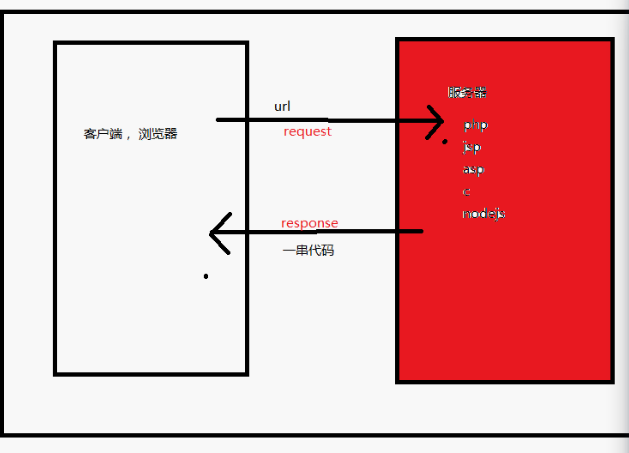
}).listen(3000);

这个方法里面是一个函数，函数有两个形参，

request ：接受客户端发来的信息，

response：返回给客户端信息。

listen()方法是服务开在哪个端口上面。localhost:3000;



当我们在node环境，开启这个模拟服务，

clipboard.png

在浏览器打开：localhost:3000,浏览器打不开任何东西，title一值在转圈，这是因为没有给客户端结束的信号，客户端一直在等待这数据的传输。所以加入response.end();就是告诉浏览器，我东西发完了，不用等了。

建立了服务器，就要用服务器给浏览器发送数据，应该如何发送？服务器会给浏览器先发送一个头（告诉浏览器以什么形式的格式去解析给它的数据）；

//请求头

response.writeHead(200,{'content-type':'text/html;charset:utf-8'});

text/html是以html的形式解析，text/plain是以文本的形式解析。

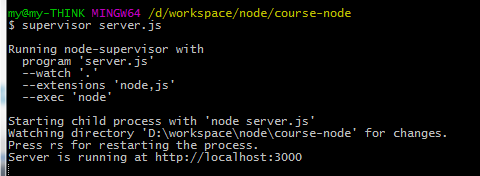
//发送给浏览器的数据

response.write('hello world~');

每次修改js文件，都要重新在node环境下运行，安装一个npm插件，能够自动去刷新。

clipboard.png

在后面就不用node server.js，直接用supervisor server.js也能打开这个模拟服务器



补充：

1.{

"name": "course-node", //这是项目名

"version": "1.0.0", //项目版本，1（大版本）.0（小版本）.0（修改的bug次数）

"description": "", //开发依赖

"main": "index.js", 主入口文件

"scripts": { //简化运行时候的指令。

"test": "echo \"Error: no test specified\" && exit 1"，

“main”:"supervisor server.js"

},

"author": "",

"license": "ISC",

"devDependencies": {

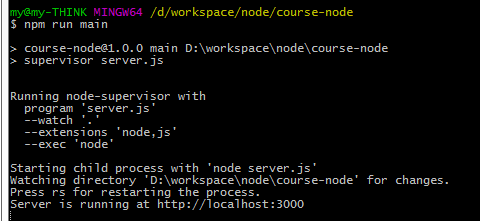
"gulp": "^3.9.1"

}

}

如果在package.json里面写了“main”:"supervisor server.js"

supervisor server.js命令就可以用 npm run main来代替。



2，为什么全局安装，还要本地安装。

因为在加载node的功能模块的时候，会先去node\_modules里面去找，不本地安装，模块的路径就要写的非常长。